НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“___”_____20 _ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки : 113 «Прикладна математика»
(код і назва)

на тему: Аналіз параметрів рекурсивних протоколів SNARK-доведень та способи симуляції доведення у цих протоколах

Виконав (-ла): студент (-ка) 4 курсу, групи ФІ-62
(шифр групи)

Стасюкевич Анатолій Тимурович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник Доктор технічних наук, професор Ковальчук Л.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Фізико-технічний інститут

Кафедра математичних методів захисту інформації

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки - 113 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою

М.М.Савчук

_____
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту

Стасюкевичу Анатолію Тимуровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз параметрів рекурсивних протоколів SNARK-доведень та способи симуляції доведення у цих протоколах,
керівник роботи доктор технічних наук, професор Ковальчук Л.В.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від _____ р. № _____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи протокол доведення без розголошення, протокол SNARK-доведень, рекурсивні доведення, протокол Coda

4. Зміст роботи під час дослідження було побудовано два способи симуляції SNARK-доведень та висунуто й доведено властивості, що необхідні для побудови тріплетів у рекурсивних SNARK-доведеннях

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	огляд опублікованих робіт та статтей за темою дослідження		
2	дослідження алгоритму побудови SNARK-доведення		
3	аналіз рекурсивних доведень		
4	побудова алгоритмів симуляції SNARK-доведень		
5	формулювання та доведення властивостей рекурсивних SNARK-доведень		

Студент _____ Стасюкевич А.Т.
(підпис) (ініціали, прізвище)

Керівник роботи _____ Ковальчук Л.В.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Кваліфікаційна робота містить: 52 стор., 2 рисунки, 7 джерел.

Метою даної кваліфікаційної роботи є аналіз параметрів SNARK-доведень та обґрунтування властивостей триплетів у рекурсивних SNARK-доведеннях.

Під час виконання роботи було побудовано два алгоритми симуляції доведення для протоколу zk-SNARK, в залежності від моменту отримання параметрів для симуляції: під час налаштування чи після. Вперше було сформульовано та доведено властивості триплетів для рекурсивних SNARK-ів. Отримані результати можна використовувати для подальшого розвитку протоколу SNARK-доведень та протоколу Coda.

БЛОКЧЕЙН, SNARK, СИМУЛЯЦІЯ ДОВЕДЕННЯ, РЕКУРСИВНІ SNARK-ДОВЕДЕННЯ

ABSTRACT

The qualification work contains 52 pages, 2 figures, 7 sources.

The goal of this qualification work is to analyze the parameters of SNARK-proofs and substantiate the properties of triplets in recursive SNARK-proofs.

During the work, two proof simulation algorithms were built for the zk-SNARK protocol, depending on when the parameters for the simulation were received: during configuration or after. For the first time, the properties of triplets for recursive SNARKs were formulated and proved. The obtained results can be used for further development of the SNARK-proof protocol and the Coda protocol.

BLOCKCHAIN, SNARK, SIMULATION OF PROOF, RECURSIVE SNARK-PROOF

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

zkp - zero-knowledge proof/protocol

nzkp - non-interactive zero-knowledge proof/protocol

nizk - non-interactive zero-knowledge

zk-SNARK- zero knowledge succinct non-interactive argument of knowledge

ppt - probabilistic polynomial time

bpp - bounded-error probabilistic polynomial time

np - nondeterministic polynomial time

r1cs - rank 1 constraints system

qap - quadratic arithmetic program

mlp - multi-party

http (англ. Hyper Text Transfer Protocol) - протокол передачі даних, що використовується в комп'ютерних мережах.

блокчейн, тобто ланцюжок блоків транзакцій — розподілена база даних, що зберігає впорядкований ланцюжок записів (так званих блоків), що постійно довшає.

криптовалюта - вид цифрової валюти, емісія та облік якої засновані на асиметричному шифруванні і застосуванні різних криптографічних методів захисту.

bitcoin - електронна валюта, концепт якої був озвучений 2008 року Сатосі Накамото, і представлений ним 2009 року, базується на самоопублікованому документі Сатосі Накамото.

гаманець - це якийсь аккаунт в платіжній системі або інет-банкінгу, де відображається наявна кількість грошей.

ВСТУП

Актуальність дослідження: питання приватності є важливим для нашого світу. Приватність даних, що відносяться до особистості людини (ідентифікаційний номер, номер паспорту, рахунок в банку, облікові записи), мають життєво необхідне значення. В наш час людина генерує дуже великий об'єм даних, як ніколи раніше, частина яких доступна всім. Великі компанії, такі як Google, Facebook, Amazon, використовують наші дані, щоб стати технологічними гігантами. Однак прориви криптографії та становлення блокчейну дають нам нові можливості захисту та ідентифікації інформації навіть від тих організацій, з якими ми взаємодіємо. Одною з найбільш популярних таких можливостей являється доведення без розголошення, а саме протокол zk—SNARK. Доведення без розголошення бере початок з 1980—х років, а протокол zk—SNARK, як різновид ZKP, вперше було згадано в 2012р. Ця технологія активно розвивається, одна із причин популярності Zcash являються саме zk—SNARKs. В 2017 році Zcash потрапив до ТОП—10 цифрових активів за рівнем капіталізації - пряме відображення популярності криптовалюти. Враховуючи таку популярність валюти, постає питання безпечного використання zk–SNARK. Одним із ключових етапів в протоколі є побудова доведення, по якому верифікатор дає відповіть так чи ні, і якщо мати таку можливість симулювати коректні доведення для zk—SNARK, можна непогано так збагатитись. Саме це питання, способів симуляції доведення, буде розглядатись у цій роботі. Окрім способів симуляції ми розглянемо також поняття рекурсивних SNARKs.

Метою даної роботи є визначити, які можливі способи симуляції для SNARK-доведень, і що для цього потрібно (мається на увазі дані). Розглянути поняття рекурсивних доведень. В часності сформулювати та довести властивості тріплетів (англ. triplets), якими володіють рекурсивні

SNARK-доведення. **Завдання даного дослідження:**

1) побудувати алгоритм симуляції доведення, для якого вхідними параметрами являються значення з налаштування SNARK протоколу;

2) побудувати алгоритм симуляції доведення, для якого вхідними параметрами являються значення, які отримав зловмисний учасник, небравший участі в налаштуванні;

3) розглянути рекурсивні доведення;

4) довести сформульовані властивості тріплетів у SNARK-доведення;

Об'єкт дослідження - процес побудови доведення у протоколі zk—SNARK та аналіз параметрів, які потрібні для цього. Також об'єктом дослідження є трійки, які необхідно задати для рекурсивних SNARKs.

Предмет дослідження - являється структура алгоритму побудови доведення для SNARK та трійки для задання рекурсивних.

Наукова новизна: вперше були розглянуті способи симуляції доведення для SNARK-доведень, в залежності від тих параметрів, що були отримані під час етапу налаштування SNARK-у та після етапу налаштування. Також в даному дослідженні вперше було сформовано та доведено властивості тріплетів для рекурсивних SNARK-доведень. Результати моєї робити можна використати для подальшого розвитку протоколу SNARK-доведень[4] та Coda protocol[7].

Практичне значення: результати даної роботи допоможуть покращити алгоритм побудови доведення для zk—SNARK з сторони безпеки та більш точно будувати основу для задання рекурсивних SNARK-доведень.

1 ДОВЕДЕННЯ БЕЗ РОЗГОЛОШЕННЯ ТА РЕКУРСИВНІ ДОВЕДЕННЯ

У даному розділі спочатку розглядаються базові поняття доведення без розголошення. Далі описуються дві форми доведення без розголошення: інтерактивна і неінтерактивна. Також в даному розділі розглядається протокол Шнорра, як приклад інтерактивної форми ZKP. У кінці приводиться приклад неінтерактивної форми ZKP, що називається zk—SNARK. zk—SNARK є ключовим об'єктом нашого дослідження, тому що дане дослідження описує саме способи симуляції для zk—SNARK.

1.1 Базові поняття

Анонімність транзакцій — одна з причин популярності криптовалют та широкого поширення технології блокчейну, але далеко не всі цифрові монети являються повністю анонімними. (Анонімна або орієнтована на конфіденційність монета — це різновид криптовалют, що забезпечують конфіденційність і анонімність своїх користувачів). Наприклад, через блокчейн можна прослідкувати транзакції в мережі Bitcoin, обчисливши адрес гаманця відправителя та отримувача, а сума переказу взагалі відома спочатку. Це ставить під удар конкуренцію криптовалют з стандартними банківськими платіжними системами. Відомий підприємець та криптоінвестор, Стів Уотерхаус (англ. Steve Waterhouse), привів вдалий приклад, він сказав, що якщо Bitcoin — це протокол передачі даних http для фінансового ринку, то Zcash — це протокол https (протокол з підтримкою шифрування задля підвищеної безпеки). Більшість криптоентузіастів погодились з даною аналогією. Zcash традиційно вважають анонімною та захищеною криптовалютою. 20 січня

2016 року було об'явлено про створення нової криптовалюти Zcash. Zcash(ZEC) — перша в світі анонімна криптовалюта з відкритим вихідним кодом на базі блокчейну, а також однойменна платіжна система. На відміну від інших анонімних цифрових активів, Zcash використовує криптографічні методи на основі доведення без розголошення, даний пункт (1.1) приводить базові поняття про нього. А також ключовою відмінністю являється протокол zk—SNARK (який є основним об'єктом нашого дослідження, та опис якого приводиться у пункті (1.4), працює на основі доказу з нульовим розголошенням, кодуючи деякі правила консенсусу в мережі. В результаті формується захищений реєстр даних без розкриття інформації про сторони і суму транзакцій. А тепер наведемо деякі визначення технології доведення без розголошення—ZKP.

Доведення без розголошення (англ. zero—knowledge proof, zero—knowledge protocol) — протокол доведення однією стороною (її називають англ. prover) істинність твердження (зазвичай математичного) іншій стороні (її називають англ. verifier) без розкриття жодної інформації, окрім достовірності твердження.

Нехай $F = \{0,1\}$, L є мовою. Нагадаємо, що $L \in NP$ означає існування детермінованого алгоритму $V : F^* \times F^* \to \{0,1\}$ і, якщо $x \in L$, означає, що існує $P \in F^*$, такий що $V(x, P) = 1$ з $|P| < poly(|x|)$. Ми кажемо, що P є коротким доведенням (англ. «short proof») того, що $x \in L$.

Позначимо $L \subseteq F^*$, тоді система доведення з нульовим розголошенням для L є пара (P, V), що задовільняє таким властивостям:

1) Повнота (англ. completeness): якщо твердження є істинним, то чесний $Prover$ завжди переконає в цьому чесного $Verifier$.

$$\forall x \in L,\ Pr[(P, V)[x] = 1] \to 1 \tag{1.1}$$

2) Коректність (англ. soundness): якщо $Prover$ — зловмисник, то ймовірність того, що $Prover$ доведе істинність твердження знехтовно

мала.

$$\forall x \notin L, \forall \hat{P} - \text{зловмисник} : Pr\left[\left(\hat{P}, V\right)[x] = 1\right] \to 0 \qquad (1.2)$$

3) Нульове розголошення (англ. zero-knowledge): якщо твердження, що тримає в секреті $Prover$, є істинним, то жодний зловмисний $Verifier$ не зможе дізнатися нічого про твердження, окрім його істинності. Система доведення (P, V) для мови L є з нульовим розголошенням, якщо для будь-якого PPT (англ. probabilistic polynomial time) $Verifier - \hat{V}$ (зловмисний $Verifier$) відомий очікуваний PPT симулятор такий що:

$$\forall x \in L, z \in \{0,1\}^* : \ View_{V^*}[P(x) \leftrightarrow V^*(x,z)] = S(x,z) \qquad (1.3)$$

(1.3) дане твердження означає, що у результаті взаємодії сторони P з зловмисним $Verifier$ V^* ніякої додаткової інформації про твердження V^* не дізнається. Отже, бесіда з $Prover$ не може навчити V^* обчислювати те, що раніше він не зміг би.

Також треба відмітити те, що ZKP не є доведенням в математичному сенсі, тому що існує знехтовно мала ймовірність (англ. soundness error) того що, нечесний $Prover$ переконає $Verifier$ в істинності хибного твердження.

Наведемо деякі означення з теорії складності (англ. complexity theory).

Введемо поняття BPP (англ. bounded-error probabilistic polynomial time) — це клас вирішення задач, що вирішуються ймовірністною машиною Тьюрінга за поліноміальний час з ймовірністю помилки $\leq 1 \div 3$ для будь—якого входу.

Неважко замітити, що якщо для мови L визначено ZKP, і у якій надсилається лише одне повідомлення, то $L \in BPP$.

У даному пункті ми описали базові поняття протоколу ZKP, що використовуються в Zcash. В наступному пункті розглянемо базову форму ZKP, яка була створена у 1989 році.

1.2 Інтерактивна форма ZKP

Доведення без розголошення бере початок з статті Гольдвассер Ш., Мікалі С., Ракофф Ч. «The knowledge complexity of interactive proof systems»[1]. Більшість робіт того часу фокусувались на системах доведення коректності, тобто на тих випадках, коли зловмисний $Prover$ старається підсунути хибне твердження $Verifier$. Але Гольдвассер, Мікалі та Ракофф розглянули протилежну сторону цієї проблеми. Замість того, щоб недовіряти $Prover$, вони спитали : що буде, якщо ми недовірятимемо $Verifier$? Особливо їх хвилювала витічка інформації. Те, що запропонували Гольдвассер, Мікалі, Ракофф стало надією на появу нових методів підтвердження.

В цій роботі[1] вчені описали ієрархію інтерактивних систем з доведенням, саме в цій статті вперше було описано ZKP. Вчені також запропонували перший приклад доведення без розголошення — доведення базувалось на знанні квадратичного лишку по заданому модулю.

В даному пункті описується інтерактивна форма ZKP. Властивості та поняття, що були приведені в пункті (1.1) стосуються саме інтерактивної форми доведення без розголошення. Також буде приведений приклад протоколу Шнорра.

Протокол Шнорра

Шнорр 1991р. запропонував наступний протокол, який є гарним прикладом інтерактивного доведення без розголошення. Доведення в даному протоколі базується на знанні дискретного логарифму. Нехай G_q — циклічна група, порядку q з генератором g. $Prover$ знає дискретний логарифм такий, що $x = log_g\ h$ й хоче довести це $Verifier$.

Алгоритм доведення для протоколу Шнорра

Публічні параметри: g , h.

Приватний параметр: x , який відомий лише стороні P $(x\ =\ log_g\ h)$.

1) $P \to V$, сторона P вибирає довільне $r \in Z_q$, і надсилає $a = g^r$.

2) $P \leftarrow V$, сторона V вибирає $b \leftarrow_R Z_q$, і надсилає b.

3) $P \to V$, сторона P надсилає $c = r + xb\,(mod\ q)$.

4) **Перевірка.** Сторона V перевіряє, що $ah^b = g^c$.

Наведемо декілька зауважень відносно протоколу Шнорра, які є необхідними для коректної роботи протоколу.

Зауваження 1. Якщо Z_q містить відносно малу кількість елементів, то протокол не є досить корисним. Оскільки протокол Шнорра є прикладом інтерактивної форми, а вона в свою чергу є імовірністною формою, для даної форми важливим являється кількість випробувань (тобто величина $|Z_q|$).

Зауваження 2. Якщо Z_q містить відносно велику кількість елементів, то протокол може неправильно працювати.

Отже, відповідно зауваженням 1 та 2, важливим є знайти компроміс для розміру $|Z_q|$.

Перша форма, що була створена, доведення без розголошення є інтерактивною. Тобто інтерактивний ZKP це протокол, що відповідає властивостям (1), (2), (3). Дана форма ZKP передбачає взаємодію між стороною, що доводить істинність твердження, та стороною, що перевіряє істинність.

Опишемо роботу інтерактивних доведень. Інтерактивні доведення характеризуються двома властивостями повнотою (англ. completeness) та коректністю (англ. soundness). Обчислення в даній формі відбувається за рахунок обміну повідомлення між двома сторонами P та V. Даний приклад (протокол Шнорра) є чудовим прикладом інтерактивного доведення, тому що передбачає спілкування між двома сторонами.

Приведемо опис одного раунду при інтерактивному доведенні, сторона, що доводить істинність твердження — P, сторона, що його перевіряє — V:

1) $P \to V$: свідчення (англ. witness)

2) $P \leftarrow V$: виклик (англ. challendge)

3) $P \rightarrow V$: відповідь (англ. response)

Як ми бачимо, спочатку P стверджує про істинність секретного твердження, не розголошуючи його. По цьому секретному твердженні обчислюється і публікується відкритий ключ. V, оцінивши публічний ключ, надсилає P виклик, деяке питання, яке перевіряє, що P дійсно знає твердження. У відповідь на питання V, сторона P надсилає відповідь на питання. Данної інформації вистачає, для того щоб V впевнився у тому, що P знає твердження. Раунди можна повторювати безліч разів, поки ймовірність того, що P «вгадує» відповіді не буде близькою до нуля.

Підсумуємо, інтерактивна форма ZKP є початковою формою даного протоколу. Інтерактивний протокол базується на раундах, чим більше раундів, тим точніше робиться аналіз знання свідчення. Ця форма є ефективною, коли в блокчейн мережі невелика кількість учасників, тому що даний процес взаємодії потрібно буде повторити знову і знову для кожного члена мережі, оскільки просто наблюдаючи він не погодиться з вами. Якщо кількість членів в мережі буде великою, то це буде проблемою. На допомогу приходить інша форма доведення без розголошення, яка буде розглянута в наступному пункті.

1.3 Неінтерактивна форма ZKP

Даний пункт розглядає форму ZKP для якої взаємодія між сторонами не потрібна, ця властивість є важливою, коли в мережі блокчейн велика кількість учасників. Саме NZKP[2] є основою для zk-SNARK. Сторона P, що доводить знання секретного твердження, відправляє лише одне повідомлення (зазвичай математичне доведення) стороні V, що легко або приймає доведення, або ні.

Система доведення NZKP[2] для деякого $x \in L$, з секретним твердженням ω, це множина ефективних (PPT) алгоритмів (K,V,P) таких що:

1) *Генератор ключів* (англ. key genarator): $\sigma \leftarrow K\left(1^k\right)$ генерує

публічну величину (англ. «public string»).

2) *Генератор доведення*: $\pi \leftarrow P(\sigma,\omega,\pi)$ сторює доведення.

3) *Алгоритм перевірки*: $\{0,1\} \leftarrow V(\sigma,x,\pi)$ приймає/відхиляє доведення.

Дані алгоритми, які задовільняють властивостям повноти, коректності та нульового розголошення, що наведені нижче.

Зауваження: Будемо вважати, що x має поліноміально обмежену довжину, тобто будемо розглядати мову $L \bigcap \{0,1\}^{P(k)}$.

Опишемо властивості для NZKP:

1) Повнота (англ. completeness). $\forall x \in L$, $\omega \in R_L(x)$:

$$Pr[\sigma \leftarrow K(1^k); \pi \leftarrow P(\sigma,\, x,\, \omega) : V(\sigma,\, x,\, \pi) = 1] = 1 \tag{1.4}$$

2) Коректність (англ. soundness).

$$Pr_{\sigma}[\sigma \leftarrow K(1^k); \exists\ (x,\, \pi) \text{ така, що } V(\sigma,\, x\ ,\pi) = 1] = 0 \tag{1.5}$$

3) Нульове розголошення (англ. zero-knowledge). Дана властивість гарантує те, що під час генерації ключів, етапу налаштування (англ. setup) ніяких даних про твердження жоден з сторони V не дізнається, окрім істиності твердження.

Саме неінтерактивна форма зараз використовується для бокчейну. Неінтерактивна форма дозволяє довести твердження великій кількості учасників мережі, тобто сторона P створила доведення, виклала його в блокчейн мережу, і будь-який чесний $Verifier$ може впевнитись в даному доведенні. Якщо аналізувати з сторони конфіденційності, то дана форма являється більш безпечною, тому що відсутність взаємодії між сторонами дає нам більшу впевненість в тому, що ніякої витічки інформації під час взаємодії не відбудеться. Інтерактивна форма потребує взаємодії сторін, але якщо одна сторона дає відповідь дуже довго, або відстань між сторонами є доволі великою, або з інших технічних проблем, постає проблема взаємодії. І рішення для цього являється NZKP. Але для

неінтерактивної форми алгоритми генерування початкових аргументів, генерування доведення є більш комплексними, тому що для протоколу ми це робимо лише раз. Давайте розглянемо найбільш ефективну та популярну реалізацію NZKP — zk-SNARK.

1.4 zk-SNARK, як приклад NZKP

Одним з найбільш вивчених, а що важливіше реалізованих являється протокол zk-SNARK[3]. Даний протокол забезпечує конфіденційність та приватність в Zcash. zk-SNARK робить перевірки більш ефективними, використовує менше даних та пам'яті під час перевірки — життєво важлива функція для блокчейну, де пам'ять та простір є дуже важливим для підтримки мережі на плаву. Команда Zcash, розуміючи важливість даної технології, активно розвиває проект. В середині 2018 року вийшло оновлення, що пришвидшило обчилення та генерацію zk-SNARK підтверджень в 5 разів. А тепер перейдемо до більш детального опису цього «магічного» протоколу.

Для початку розшифруємо акронім у назві пункту, він означає англ. *zero-knowledge Succinct Non-Interactive Argument of Knowledge*. Нульове розголошення у цій назві означає, що zk-SNARK дотримується протоколу ZKP, тобто сторона P, доводить істинність твердження стороні V, без розголошення самого твердження. Стислий (англ. succinct) доказ в zk-SNARK означає, що верифікувати доказ можна за мілісекунди, займає він лише кілька сотень байтів. Non-Interactive означає, що сторона P надсилає стороні V лише один доказ, тобто взаємодії між ними немає, ця форма ZKP описана у попередньому пункті. Перейдемо до «аргументів», як частини даного акроніму. zk-SNARK являється обчислювано—обгрунтованою технологією, а це означає, що зловмисний $Verifier$ має дуже низький шанс обхитрити систему. Ця властивість передбачає те, що сторона V має обмежену обчислювану потужність.

zk-SNARK складається з трьох ефективних алгоритмів (G,P,V):

1) *Генератор ключів* G: отримує на вхід секретний параметр λ і програму c і генерує два загальнодоступні ключі (англ. publicly available keys), p_k ключ для доведення (англ. proving key) та v_k ключ для верифікації (англ. verification key). Дані ключі являються публічними і генеруються лише один раз для заданої програми.

2) *Алгоритм побудови доведення* P: отримує на вхід ключ доведення p_k, публічний параметер x та секретний параметер ω (англ. witness). Алгоритм генерує доведення $\pi = P\,(p_k,x,\omega)$.

3) *Verifier* V: отримує на вхід ключ v_k, публічний параметер x, та доведення π, після обчислення повертає 1, якщо доведення коректне, 0 - якщо інакше.

Хочу також підкреслити увагу на секретному параметрі λ. Зазвичай той, хто знає цей параметр може сформулювати доведення, що приведені в (2) пункті. Тому в реальних додатках, цьому параметру приділяється особлива увага. В наступних розділах буде на цьому наголошено. Найбільш ефективним zk-SNARK вважається запропонований Гроссом[4], який побудований для QAP[5] та працює в білінійних групах, саме його ми будемо використовувати у подальших розділах.

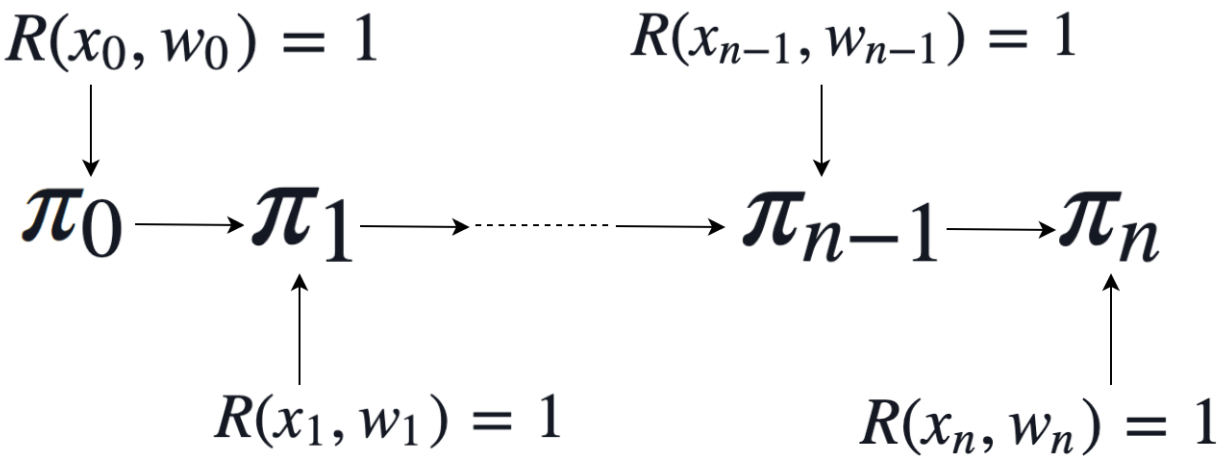
1.5 Рекурсивні доведення

Для рекурсивної системи важливим є поняття «Incremental Verifiable Computation», під час побудови доведення, ми обов'язково враховуємо попередні обчислення. Нехай R , $V \in NP$. Функція V— функція перевірки доведення, відносно протоколу SNARK-доведень це $Verifier$. Опишемо «прості» приклади:

– «Proof of a proof»: побудоване доведення π_1 свідчить про знання доведення π_0, таке що для публічного входу x, секретного твердження ω $\pi_1 \leftarrow P(x, \omega)$ (алгоритм побудови доведення) і $V(x, \pi_0) = 1$

– «Proof of a proof of a proof...» побудоване доведення π_2 свідчить про знання π_1, що перевіряє доведення π_0;

Таким чином, ми отримуємо систему рекурсивних доведень. Описується це наступною схемою.



В загальних системах доведення $Prover$ створює доведення π, що він знає деякий публічний вхід x, та секретний вхід ω, для деякого NP—відношення R, заданого арифметичною схемою, тобто π свідчить про те, що $R(x,\omega) = 1$. Потім $Verifier$ перевіряє, що π валідний, при цьому він поверне 1, якщо це так. Ця перевірка може бути виражена, як NP—відношення R', таке що $R'(x,\pi) = V(x,\pi)$ для всіх валідних входів верифікатора. Таким чином, ми можемо створювати докази, які засвідчують коректність інших доказів.

У найпростішому випадку рекурсивні докази індуктивно докажуть коректність відношення R. Іншими словами, у нас буде «базове» доведення π_0, що свідчить про те, що $Prover$ знає деякий вхід (x_0,ω_0), такий що $R(x_0,\omega_0) = 1$. Тоді доведення π_n(n>0) доведе, що $Prover$ знає (x_n,ω_n), такий що $R(x_n,\omega_n) = 1$ і π_{n-1} було засвідчено знання (x_{n-1},w_{n-1}).

Як можна це зробити? Спочатку будується схема C_V для $Verifier$-а. Тоді будується схема C за домогою якої ми легко перевіряємо $R(x_0,\omega_0) = 1$ (для базового випадку) або $R(x_i,\omega_i) = 1$, а потім перевіряє, що $V(x_i,\pi_{i-1}) = 1$. Описана вище процедура називається рекурсивним доведенням. Відповідно до рекурсивних SNARK-доведень, рекурсія доведення використовується, коли ми не викладаємо елементи груп, які

були побудовані під час алгоритму (2), а доводимо їх знання. Звідси випливає потреба у двох різних тріплетах: один для створення доведення, а інший для валідації. Вперше поняття рекурсивних SNARK-доведень було описано в протоколі Coda[7]. Але в даному протоколі не було сформульовано вимог до властивостей тріплетів, що потрібні для протоколу. Отже, одним з головних завдань мого дослідження являється: формулювання та доведення властивостей тріплетів для рекурсивних SNARK-доведень.

Висновки до розділу 1

В даному розділі розглянуто протокол доведення без розголошення та рекурсивні доведення. Було описано дві форми ZKP, а також приведені приклади для них. В кожному пункті було описано переваги та недоліки використання тої чи іншої форми. Також було приведено опис зв'язку протоколу доведення без розголошення з блокчейн технологією. Ми розглянули протокол zk-SNARK, як практичну реалізацію ZKP. zk-SNARK - є дуже перспективною технологією, оскільки навідмінно від Bitcoin, вона гарантує приватність транзакцій, переведень, рахунків. Користувач тепер може більше довіряти додатку, що побудований на цій технології. Саме zk-SNARK дають новий подих для блокчейну та криптовалют у боротьбі з банківськими системами. Конфіденційність тепер не є проблемою для блокчейну з використанням zk-SNARK. Наступні компанії вже застосували zk-SNARK у своїх проектах:

– QED-it - ізраїльська компанія, що використовує SNARK для аудиту фінансових установ;

– NuCypher - проект ICO, який спеціалізується на повторному шифруванні проксі;

– Nuggets забезпечує можливість захисту даних інтернет-магазинів;

Отримані знання у першому розділі допоможуть нам описати більш детально побудову zk-SNARK у другому розділі.

2 ПОБУДОВА ZK-SNARK

У даному розділі буде детально описана побудова zk-SNARK. Спочатку буде описано перехід до QAP рівняння, яке є ключовим для zk-SNARK. Багато уваги буде приділено етапу налаштування (англ. setup), як одного з ключових етапів побудови. Також буде розглянуто багаторазові обчислення (англ. multi-party calculations). Після етапу налаштування ми покажемо, як будується доведення (англ. proof) в zk-SNARK. Алгоритм побудови є важливим, оскільки він безпосередньо впливає на побудову симуляції доведення. Опишемо коректність (англ.correctness) та верифікацію доведення.

2.1 Перехід від R1CS до QAP

На високому рівні zk-SNARK починають роботу з того, що перетворюють свідчення, що хоче довести *Prover* в еквіваленту форму алгербаїчних рівнянь. І наступним кроком є приведення до QAP.

Алгоритм побудови доведення для zk-SNARK не можна відтворити без використання QAP. Для нього також можливі передобчислення. В даному пункті ми розглянемо, як перейти з схеми R1CS до QAP.

Покладемо p - велике просте число, F_p - скінчене поле. Візьмемо елементи поля: $u_{i,q}$, $v_{i,q}$, $w_{i,q} \in F_p$, де $i = \overline{0, m}$, $q = \overline{0, n}$. Наша мета: довести, що ми знаємо (або відомі) такі a_0, a_1,..., $a_m \in F_p$ (з $a_0 = 1$), що виконуються наступні рівняння:

$$\sum_{i=0}^{m} a_i \; u_{i,q} \; \cdot \sum_{i=0}^{m} a_i \; v_{i,q} = \sum_{i=0}^{m} a_i \; w_{i,q}, \text{ де } q = \overline{1,n} \tag{2.1}$$

Дану систему (2.1) називають (англ.) «Rank 1 Constraints System», або R1CS. Спочатку покажемо, як можна (2.1) перетворити в одне поліноміальне рівняння. Перепишемо нашу мету мовою поліномів.

Побудуємо $3\,(m+1)$ поліномів:

$$u_i(x),\, v_i(x),\, w_i(x) \in F_p[X],\, i = \overline{0,m} \tag{2.2}$$

такі, що для деякого визначеного набору r_1, ... , r_m (зазвичай $r_q = q$, $q = \overline{1,\, n}$) задані поліноми отримують відповідні значення:

$$u_i(r_q) = u_{i,q}\ ,\ v_i(r_q) = v_{i,q}\ ,\ w_i(r_q) = w_{i,q}\ ,\ i = \overline{0,\, m}\ ,\ q = \overline{1,\, n} \tag{2.3}$$

Для спрощення, ми вважатимемо $r_q = q$, $q = \overline{1,\, n}$, рівняння (2.3) можна переписати наступним чином:

$$u_i(q) = u_{i,q}\ ,\ v_i(q) = v_{i,q}\ ,\ w_i(q) = w_{i,q}\ ,\ i = \overline{0,\, m}\ ,\ q = \overline{1,\, n} \tag{2.4}$$

Для побудови поліномів (2.2) з обмеженнями (2.4), використаємо формулу Лагранжа[6], кожен поліном (2.2) має степінь $\leq n-1$.

Завдяки нашій побудові, ми зможемо сказати, що (2.1) виконується, якщо виконуються наступні рівняння:

$$\sum_{i=0}^{m} a_i\ u_i(x)\ \cdot \sum_{i=0}^{m} a_i\ v_i(x) = \sum_{i=0}^{m} a_i\ w_i(x),\ \text{де}\ x \in \{1,2,...,n\}\,. \tag{2.5}$$

Тепер нашою метою є довести, що відомі такі a_0, a_1,..., $a_m \in F_p$, що виконуються рівняння (2.5) для $x \in \{1,2,...,n\}$.

Визначимо такий поліном:

$$g(x) = \sum_{i=0}^{m} a_i\ u_i(x)\ \cdot \sum_{i=0}^{m} a_i\ v_i(x) - \sum_{i=0}^{m} a_i\ w_i(x) \tag{2.6}$$

Відповідно до (2.5) і (2.6), елементи $\{1,2,...,n\}$ є коренями полінома $g\,(x)$. За теоремою Безу це еквівалентно до:

$$g(x) \vdots (x-q)\ ,\ q = \overline{1,n}. \tag{2.7}$$

Всі поліноми $(x-q)$, $q = \overline{1,\, n}$ являються незвідними і попарно простими, відовідно до першого наслідку алгоритму Евкліда, (2.7) можна переписати,

як:

$$g(x) \vdots \prod_{q=1}^{n}(x-q) \tag{2.8}$$

Обозначимо $t\,(x) = \prod_{q=1}^{n}(x-q)$. Тепер ми можемо переписати (2.8), як $g\,(x) \vdots t\,(x)$.

Зараз наша мета довести, що існуть такі a_0, a_1,..., $a_m \in F_p$, що

$$q(x) \vdots t(x) \text{ , де } g(x) = \sum_{i=0}^{m} a_i\ u_i(x)\ \cdot \sum_{i=0}^{m} a_i\ v_i(x) - \sum_{i=0}^{m} a_i\ w_i(x) \tag{2.9}$$

Умова (2.9) може бути переписана, як:

$$\sum_{i=0}^{m} a_i\ u_i(x)\ \cdot \sum_{i=0}^{m} a_i\ v_i(x) - \sum_{i=0}^{m} a_i\ w_i(x) \equiv 0 (mod\ t(x))$$

або

$$\sum_{i=0}^{m} a_i\ u_i(x)\ \cdot \sum_{i=0}^{m} a_i\ v_i(x) \equiv \sum_{i=0}^{m} a_i\ w_i(x) (mod\ t(x))$$

Отже, ми звели наше початкове положення до наступного (англ. Constructing Non-Interactive Zero Knowledge Arguments for Quadratic Arithmetic Program (NIZK Arguments for QAP)):

Довести, що відомі a_0, a_1,..., $a_m \in F_p$ такі, що:

$$\sum_{i=0}^{m} a_i\ u_i(x)\ \cdot \sum_{i=0}^{m} a_i\ v_i(x) \equiv \sum_{i=0}^{m} a_i\ w_i(x) (mod\ t(x)),$$

$$\text{де } t(x) = \prod_{q=1}^{n}(x-q). \tag{2.10}$$

І конгуренцію, яка описана в (2.10), ми будемо називати (англ.) Quadratic Arithmetic Program (QAP).

В теорії будь-яку задачу, твердження можна звести до арифметичної схеми, що в подальшому можна буде перетворити в рівняння (2.10). А це означає універсальність zk-SNARK.

Отже, ми розглянули перехід до QAP рівняння. Треба зауважити те, що протокол zk-SNARK не може бути застосований до будь-якої обчислювальної задачі, спочатку цю задачу треба привести до правильної форми. Правильна форма для zk-SNARK — це QAP. Тепер перейдемо до початкового етапу zk-SNARK.

2.2 Опис етапу налаштування в zk-SNARK

В даному пункті буде розглянуто етап налаштування. Його також називають довіреним налаштуванням (англ. trusted setup). Найпростіший спосіб згенерувати публічні параметри — зробити це так, щоб одна довірена сторона це зробила. Але це є небезпечно та критично, оскількі всім відома приказка «довіряй, але перевіряй». Тому етап налаштування є досить складним, і в якому бере участь декілька сторін, кожна з яких контролює дії інших.

Для zk-SNARK важливим є етап налаштування публічних параметрів, які потрібні для побудови доведення та для перевірки. Важливою умовою є видалення секретних параметрів, що беруть участь у налаштуванні, після завершення етапу.

Ми описали нашу мету в пункті (2.1), використавши конгуренцію (2.10). В цьому пункті ми почнемо конструювати NIZK аргументи. Перша частина побудови zk-SNARK називається налаштуванням (англ. setup). В налаштуванні відбуваютсья всі передобчислення, які потрібні для доведення (англ. proving) та верифікації (verification).

Використовуючи MLP обчислення, учасники визначають певні змінні, які потім будуть використані стороною P та стороною V.

Деякі з цих значень відомі тільки одному з учасників, інші значення іншим учасникам. В даному пункті буде детально описано, як використовуються MLP обчислення.

Зауважимо що, в загальному, деякі змінні a_0, a_1,..., $a_m \in F_p$ можуть бути також відомі всім учасникам. Без втрати загальності, ми

припускаємо, що a_0, a_1,..., $a_l \in F_p$ відомі (також може бути, що $l = 0$) і a_{l+1}, a_{l+2},..., $a_m \in F_p$ відомі лише $Prover$. Значення a_0, a_1,..., $a_l \in F_p$ називаються «заявами» (англ. «statements»), значення a_{l+1}, a_{l+2},..., $a_m \in F_p$ називаються «свідченнями» (англ. «witnesses»). $Prover$ повинен довести, що він знає такі свідчення, що для заданих заяв, заданих поліномів $u_i(x), v_i(x), w_i(x)$, $i = \overline{0,m}$ та заданого $t(x) = \prod_{q=1}^{n}(x-q)$ виконується конгуренція (2.10).

Далі ми будемо використовувати визначення та позначення з пункту (2.1). Ми також використовуватимемо деякі позначення з [4], але деякі наші позначення відрізняються.

Наведемо опис деяких об'єктів для подальшого розгляду. Введемо трійку, $\mathbf{TR} = (\mathbf{G_1}, \mathbf{G_2}, \mathbf{G_{TG}})$, що складається з трьох груп $\mathbf{G_1}, \mathbf{G_2}, \mathbf{G_{TG}}$, які відповідають таким властивостям:

– $|\mathbf{G_1}| = |\mathbf{G_2}| = |\mathbf{G_{TG}}| = p$, де p велике просте число з частини (2.1);

– функцію спарювання, визначимо наступним чином $e : \mathbf{G_1} \times \mathbf{G_2} \to \mathbf{G_{TG}}$ — це білінійна карта (англ. bilinear map);

– $G_1 \in \mathbf{G_1}$ генератор групи $\mathbf{G_1}$, $G_2 \in \mathbf{G_2}$ генератор групи $\mathbf{G_2}$ і $G_{TG} = e(G_1, G_2) \in \mathbf{G_{TG}}$ генератор групи $\mathbf{G_{TG}}$.

Відповідно до [4] «це є ефективні алгоритми для обчислення групових операцій, обчислення білінійної карти, вирішення про належність до групи, вирішення рівності групових елементів та генераторів груп». Наведемо декілька зауважень про трійку $\mathbf{TR}$.

Зауваження 1. На сьогодні існує лише один тип для відповідних груп $\mathbf{G_1}$, $\mathbf{G_2}$ та $\mathbf{G_{TG}}$: $\mathbf{G_1}$ підгрупа еліптичної кривої $E(F_r)$ над простим полем F_r; $\mathbf{G_2}$ підгрупа еліптичної кривої $E(F_{r^k})$ над розширенням поля F_{r^k}; і $\mathbf{G_{TG}}$ підгрупа $F^*_{r^k}$.

Підгрупи $\mathbf{G_1}$, $\mathbf{G_2}$ повинні мати наступні властивості:

1) $\mathbf{G_1} \bigcup \mathbf{G_2} = \{O\}$, де O являється нейтральним елементом еліптичної кривої $E(F_{r^k})$;

2) $e : \mathbf{G_1} \times \mathbf{G_2} \to \mathbf{G_{TG}}$ - спарювання на еліптичних кривих.

Зауваження 2. Постає питання: чому ми вимагаємо, щоб порядок груп

$\mathbf{G_1}$, $\mathbf{G_2}$ та $\mathbf{G_{TG}}$ був рівний характеристиці поля F_p , де F_p поле над яким ми будували R1CS в частині (2.1) ?

Це зазначено в частині (2.1), наша мета побудувати zk-SNARK, щоб довести знання таких a_0, a_1,..., $a_m \in F_p$ щоб виконувалась конгуренція (2.10). В zk-SNARK нам потрібно перемножити елементи a_0, a_1,..., $a_m \in F_p$ на елементи $\mathbf{G_1}$, $\mathbf{G_2}$. Результат таких перемножень є також елементи $\mathbf{G_1}$, $\mathbf{G_2}$ відповідно. Для прикладу $a \in F_p$, $G \in \mathbf{G_1}$: ми маємо $a \cdot G \in \mathbf{G_1}$. Ми хочемо, щоб F_p та G_1 були сумісні в наступному сенсі : ми вимагаємо дотримуватись «природніх» (англ. «natural») законів для множення. Ми вимагаємо:

– $\forall\ a\ , b \in F_p$ і $G \in \mathbf{G_1}$:

$$aG\ +\ bG = (a+b)\ G; \tag{2.11}$$

– якщо $0 \in F_p$ - нейтральний елемент F_p , $G \in \mathbf{G_1}$, $O \in \mathbf{G_1}$ - нейтральний елемент тоді отримуємо:

$$0 \cdot G\ =\ O; \tag{2.12}$$

А зараз змоделюємо ситуацію. Нехай $\forall\ a\ , b \in F_p : a = p - b$. В F_p це означає, що $a + b = 0$. Відповідно до (2.11) та (2.12) виконуються наступні рівняння:

$$aG_1 + bG_1 = (a+b)G_1 = 0 \cdot G_1 = O.$$

Але відповідно до наслідку теореми Лагранжа[6], рівність:

$$(a+b)\ G_1\ =\ O$$

означає $a+b \vdots ordG_1$, або $p \vdots\ |\mathbf{G_1}|$ (тому що $a\ +\ b\ =\ p$, якщо ми вважаємо a та b не елементами поля, але натуральними числами і $ordG_1 = |\mathbf{G_1}|$). Але p та $|\mathbf{G_1}|$ це прості числа, тому $p \vdots\ |\mathbf{G_1}|$ еквівалентно $p = \mathbf{G_1}$.

Продовжимо вивчення NIZK (або zk-SNARK) запропоноване в [4].

Беручи до уваги те, що групи, як правило, це групи точок еліптичних кривих (точніше, підгрупи простих порядків деяких еліптичних кривих). Ми будемо використовувати «+» для операцій у заданих групах (це буде зрозуміло з контексту, в яких групах ми працюємо). $\mathbf{G_{TG}}$ це , зазвичай, підгрупа мультиплікативної групи скінченого(або розширення) поля, ми будемо використовувати «∗» для операцій в $\mathbf{G_{TG}}$.

Нижче ми наводимо повні NIZK аргументи, що описані в [4].

$(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$: Pick $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{F}^*$. Set $\boldsymbol{\tau} = (\alpha, \beta, \gamma, \delta, x)$ and

$$\boldsymbol{\sigma} = \left(\alpha, \beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}, \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right\}_{i=0}^{\ell}, \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}\right\}_{i=\ell+1}^{m}, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n-2} \right).$$

$\boldsymbol{\pi} \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, a_1, \ldots, a_m)$: Pick $r, s \leftarrow \mathbb{F}$ and compute a $3 \times (m + 2n + 4)$ matrix Π such that $\boldsymbol{\pi} = \Pi\boldsymbol{\sigma} = (A, B, C)$ where

$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta \qquad B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta$$

$$C = \frac{\sum_{i=\ell+1}^{m} a_i \left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) + h(x)t(x)}{\delta} + As + rB - rs\delta.$$

$0/1 \leftarrow \mathsf{Vfy}(R, \boldsymbol{\sigma}, a_1, \ldots, a_\ell)$: Compute a quadratic multi-variate polynomial $\boldsymbol{t}$ such that $\boldsymbol{t}(\boldsymbol{\sigma}, \boldsymbol{\pi}) = 0$ corresponds to the test

$$A \cdot B = \alpha \cdot \beta + \frac{\sum_{i=0}^{\ell} a_i \left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right)}{\gamma} \cdot \gamma + C \cdot \delta.$$

Accept the proof if the test passes.

Рисунок 2.1 – опис NZIK аргументів в [4]

Для простоти подальшого розуміння розділимо вектор елементів на два вектори:

$$\sigma_1 = (\ \alpha\ ,\ \beta\ ,\ \delta\ ,\ \{x^i\}_{i=0}^{n-1}\ ,\ \{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\}_{i=0}^{l}\ ,$$
$$\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}\}_{i=l+1}^{m}\ ,\ \{\frac{x^i t(x)}{\delta}\}_{i=0}^{n-2})$$

$$\sigma_2 = (\ \beta\ ,\ \gamma\ ,\ \delta\ ,\ \{x^i\}_{i=0}^{n-1}\ )$$

Як ми можемо побачити лише декілька слів у 2.1 про налаштування NIZK аргументів. Але цей опис викликає багато питань:

1) Хто задає параметри α , β , γ , δ та «toxic waste» x ?

2) Всі учасники знають ці значення, або деякі з них, або ніхто?

3) Якщо значення α , β , γ , δ , x є невідомі, як хтось може обрахувати $[\sigma_1]_1 = \sigma_1 G_1$ та $[\sigma_2]_2 = \sigma_2 G_2$?

4) Якщо значення відомі кожному, то він зможе обрахувати симульоване доведення для $\forall A$, $B \in Z_p$ - тому вони повинні бути невідомі?

Зараз ми постараємось відповісти на ці запитання. Головна ідея етапу налаштування в тому, що всі значення

$$\alpha\ ,\ \beta\ ,\ \gamma\ ,\ \delta \text{ та «toxic waste» } x \tag{2.13}$$

рахуються, використовуючи MLP обчислення. Учасники не отримують значення (2.13) собі, і ніхто не знає цих значень. Але вони оцінюють елементи групи, такої форми

$$\alpha G_1\ ,\ \beta G_1\ ,\ \delta G_1\ ,\ ... \in \mathbf{G_1} \text{ та } \beta G_2\ ,\ \gamma G_2\ ,\ \delta G_2\ ,\ ... \in \mathbf{G_2}$$

не знаючи значення (2.13). Тепер пояснимо, як вони це роблять, припускаючи для простоти, що є лише три учасники: $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$. Ми почнемо з простого алгоритму для отримання αG_1 .

Тож, ми припускаємо, що є три сторони ($\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$) і кожна з них знає генератори $G_1 \in \mathbf{G_1}$ та $G_2 \in \mathbf{G_2}$. Вони хочуть побудувати деяку точку $G_\alpha^{(1)} = \alpha G_1 \in \mathbf{G_1}$, таку що:

– ніхто з них не знає α;

– кожен з них знає αG_1;

– кожен з них бере участь у розрахунку αG_1;

– якщо зловмисник брав участь у побудові αG_1, то всі сторони це побачать.

Опишемо простий алгоритм для побудови αG_1 .

Алгоритм №1

Спрощений алгоритм для побудови $G_\alpha^{(1)} = \alpha G_1$

1) сторона $\mathbf{P_1}$ генерує деяке рандомне $\alpha_1 \in F_p^*$, обраховує $U_1 = \alpha_1 G_1$ і викладає U_1 в блокчейн ;

2) сторона $\mathbf{P_2}$ генерує деяке рандомне $\alpha_2 \in F_p^*$, обраховує $U_{12} = \alpha_2 U_1$ і викладає U_{12} в блокчейн ;

3) сторона $\mathbf{P_3}$ генерує деяке рандомне $\alpha_3 \in F_p^*$, обраховує $U_{123} = \alpha_3 U_{12}$ і викладає U_{123} в блокчейн ;

4) зафіксуємо $G_\alpha^{(1)} = U_{123}$;

5) викладаємо $G_\alpha^{(1)}$ в блокчейн.

Цей алгоритм достатній, коли ми впевнені, що всі сторони $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$ є чесними. В протилежному випадку нам потрібно добавити деяку перевірку, щоб бути впевненим, що всі сторони знають відповідні їм значення α_1 , α_2 , $\alpha_3 \in F_p^*$ і не викладають довільні вибрані точки U_1 , U_2 , U_3 , замість їх обчислення відповідно до **Алгоритм №1** .

Тому нам потрібно певні перевірки, які ми наведемо у наступному алгоритмі.

Наведемо означення проблеми CDH (англ. computational Diffie–Hellman). Нехай G — це циклічна група порядку q, g — генератор групи. Виникає наступна проблема (CDH):

Дано: g , g^a , g^b , де a , $b \in \{0,...,q-1\}$.
Знайти: g^{ab}.

Алгоритм №2

Алгоритм для побудови $G_\alpha^{(1)} = \alpha G_1$ з перевіркою

1) Кожна сторона $\mathbf{P_i}$, $i = \overline{1,3}$ вибирає деяке рандомне значення α_i , $i = \overline{1,3}$, обчислює $U_i = \alpha_i G_1$ та $T_i = \alpha_i G_2$, $i = \overline{1,3}$ і викладає U_i та T_i в блокчейн;

2) Кожна сторона перевіряє, що наступне рівняння виконується

$$e(U_i \ , G_2) = e(G_1 \ , T_i) \quad , i = \overline{1,3} \tag{2.14}$$

Зауваження. Якщо $\mathbf{P_i}$ є чесним, то виконується наступна рівність:

$$e(U_i \ , G_2) = e(\alpha_i G_1 \ , G_2) = e(G_1 \ , \alpha_i G_2) = e(G_1 \ , T_i) \ , i = \overline{1,3}$$

Інакшими словами, рівняння (2.14) перевіряє, що $\mathbf{P_i}$ знає α_i і обчислює U_i та T_i коректно, з тим α_i. Припускаючи, що проблема CDH є важкою, дана перевірка є достатньою. Якщо $i-$ те рівняння не виконується, то сторона $\mathbf{P_i}$ є зловмисником.

3) Сторона $\mathbf{P_2}$ обчислює $U_{12} = \alpha_2 U_1$ і викладає U_{12} в блокчейн;

4) Кожна сторона перевіряє чи виконується наступне рівняння:

$$e(U_{12} \ , G_2) = e(U_1 \ , T_2) \tag{2.15}$$

Перевірка. Якщо (2.15) не виконується, то сторона $\mathbf{P_2}$ є зловмисником.

5) Сторона $\mathbf{P_3}$ обчислює $U_{123} = \alpha_3 U_{12}$ і викладає U_{123} в блокчейн;

6) Кожна сторона перевіряє, що виконується наступне рівняння:

$$e(U_{123} \ , G_2) = e(U_1 2 \ , T_3) \tag{2.16}$$

Перевірка. Якщо (2.16) не виконується, то сторона $\mathbf{P_3}$ є зловмисником.

7) Якщо 2.14 - 2.16 виконуються, то фіксуємо:

$$G_\alpha^{(1)} = U_{123}$$

Якщо хоча б одне з рівнянь 2.14 - 2.16 не виконується, тоді учасники повинні виключити зловмисника та перезапустити протокол. Або

зловмисники повинні бути покарані.

8) Викладаємо $G_\alpha^{(1)}$ в блокчейн.

Зауваження. Алгоритм №2 є спрощеним у порівнянні з реальною програмою, але він показує головну ідею MLP обчислень в налаштуванні (англ. setup).

Для побудови значень в налаштуванні (англ. setup), щоб використати **Алгоритм №2**, кожна сторона повина спочатку створити та викласти наступні значення:

$$\alpha_i G_1 \ , \ \beta_i G_1 \ , \ \delta_i G_1 \ , \ x_i G_1 \ , \ \beta_i G_2 \ , \ \gamma_i G_2 \ , \ \delta_i G_2 \ , \ x_i G_2.$$

Створюємо їх, використовуючи значення α_i , β_i , δ_i , x_i , γ_i , які вибирають сторони рандомно з F_p^* і тримають у секреті.

Використовуючи **Алгоритм №2** і викладені значення, ми можемо побудувати наступні «прості» значення з налаштування:

– $G_\alpha^{(1)} = \alpha G_1$;

– $G_\beta^{(1)} = \beta G_1$;

– $G_\delta^{(1)} = \delta G_1$;

– $\left\{X_i^{(1)} = x^i G_1\right\}_{i=0}^{n-1}$ (з додатковою перевіркою, що при побудові $X_{i+1}^{(1)}$ з $X_i^{(1)}$ сторона P_j множить $X_i^{(1)}$ на такий самий x_j , що використовується в xG_1) ;

– $G_\beta^{(2)} = \beta G_2$;

– $G_\gamma^{(2)} = \gamma G_2$;

– $G_\delta^{(2)} = \delta G_2$;

– $\left\{X_i^{(2)} = x^i G_2\right\}_{i=0}^{n-1}$ (з додатковою перевіркою, що при побудові $X_{i+1}^{(2)}$ з $X_i^{(2)}$ сторона P_j множить $X_i^{(2)}$ на такий самий x_j , що використовується в xG_2 ; і також слід перевірити, що кожна сторона P_j використовує в xG_2 той самий x_j, що і в xG_1) ;

Після побудови цих «простих» змінних, ми повинні побудувати більш скадніші:

$$\{G_{\alpha\beta\gamma,i}^{(1)} = \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} G_1\}_{i=0}^{l} \tag{2.17}$$

$$\{G^{(1)}_{\alpha\beta\delta,i} = \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} G_1\}_{i=l+1}^{m} \tag{2.18}$$

де x це «toxic waste».

Зараз ми наведемо алгоритм для обчислення (2.17). Для (2.18) алгоритм є такий самий, тільки потрібно замінити γ на δ.

Спочатку нагадаємо, що всі поліноми $u_i(x)$, $v_i(x)$ та $w_i(x)$ відомі всім учасникам (іншими словами, коефіцієнти цих поліномів відомі). І також всі учасники знають всі значення $\left\{X_i^{(1)} = x^i G_1\right\}_{i=0}^{n-1}$. Зараз ми покажемо, як учасники обраховують $u_i(x)G_1$.

Нехай $u_j(x) = f_{n-1}^{(j)} x^{n-1} + f_{n-2}^{(j)} x^{n-2} + ... + f_1^{(j)} x + f_0^{(j)}$, $j = \overline{0,m}$.

Тоді визначимо:

$$\sum_{i=0}^{m-1} f_i^{(j)} X_i^{(1)} = \sum_{i=0}^{m-1} f_i^{(j)} x^i G_1 = (\sum_{i=0}^{m-1} f_i^{(j)} x^i) G_1 = u_j(x) G_1 \ , \ j = \overline{0,m}. \tag{2.19}$$

Обрахуємо $\beta u_j(x) G_1$ знаючи $u_j(x)G_1$, отримане відповідно до (2.19), учасники послідовно перемножують $u_j(x)G_1$ на β_1 (учасник $\mathbf{P_1}$) , потім на β_2 (учасник $\mathbf{P_2}$) , і під кінець на β_3 (учасник $\mathbf{P_3}$) , разом з відповідними підтвердженнями , що змінні β_1 , β_2 , β_3 є однаковими в $G_\beta^{(1)}$ та $G_\beta^{(2)}$.

Таким самим чином учасники обчислюють значення

$$\alpha v_i(x) G_1 \text{ та } w_i(x) G_1$$

а потім додають їх, $j = \overline{0,m}$. Таким чином ми знайшли суму $(\beta u_i(x) + \alpha v_i(x) + w_i(x))G_1$, $j = \overline{0,m}$, і потім множим цю суму на γ^{-1} (для $j = \overline{0,l}$) або на δ^{-1} (для $j = \overline{l+1,m}$). Для цих перемножень можна використати **Алгоритм №2** , але з деякими виправленнями: після того, як $\mathbf{P_i}$ фіксує відповідний γ_i (або δ_i) , він знаходить $\gamma_i^{-1} mod p$ (або $\delta_i^{-1} mod p$) і потім множить відповідне значення на $\gamma_i^{-1} mod p$ замість γ_i (або $\delta_i^{-1} mod p$ замість δ_i) . І, звичайно, з обов'язковими перевіркаим після кожного множення. Після цього значення (2.17) та (2.18) знайдені.

Щоб завершити налаштування, учасники повинні знайти значення

$$\{T_{\delta,i}^{(1)} = \frac{x^i t(x)}{\delta} G_1\}_{i=0}^{n-2}.$$

Нагадаємо, що поліном

$$t(y) = \prod_{i=1}^{n} (y - i).$$

відомий всім сторонам, вони також знають поліноми $y^i t(y)$ і можуть знайти значення $\{x^i t(x) G_1\}_{i=0}^{n-2}$, так само, як було знайдено $u_j(x)G_1$ в (2.19). Потім учасники множать $x^i t(x) G_1$ на δ_1^{-1} (учасник $\mathbf{P_1}$), потім на δ_2^{-1} (учасник $\mathbf{P_2}$) і в кінці на δ_3^{-1} (учасник $\mathbf{P_3}$) з обов'язковою перевіркою після кожного множення і отримуємо значення

$$\{T_{\delta,i}^{(1)} = \frac{x^i t(x)}{\delta} G_1\}_{i=0}^{n-2}.$$

Тепер учасники отримали (обчислили) всі параметри для налаштування.

Для зручності подальших викладок, перепишемо отримані значення з етапу налаштування:

$$\begin{aligned} \sigma_1 G_1 = \{&G_\alpha^{(1)} \ , G_\beta^{(1)} \ , G_\delta^{(1)} \ , \{X_i^{(1)}\}_{i=0}^{n-1} \ , \\ &\{G_{\alpha\beta\gamma,i}^{(1)}\}_{i=0}^{l} \ , \{G_{\alpha\beta\delta,i}^{(1)}\}_{i=l+1}^{m} \ , \{T_{\delta,i}^{(1)}\}_{i=0}^{n-2}\} \end{aligned} \tag{2.20}$$

$$\sigma_2 G_2 = \{G_\beta^{(2)} \ , G_\gamma^{(2)} \ , G_\delta^{(2)} \ , \{X_i^{(2)}\}_{i=0}^{n-1} \ \} \tag{2.21}$$

Отже, в даному пункті ми розглянули налаштування публічних параметрів для подальшої побудови доведення та верифікації. Великі проекти по типу Zcash знають проблему в налаштування, це токсичні параметри. Zcash цілком конфіденційно відноситься до етапу налаштування.

Перша церемонія відбулась в жовтні 2016 року, в ній участь прийняли 6 відомих розробників в області криптовалют. Даний протокол дає достатню гарантію безпеки, якщо хоча б один з учасників буде

чесним, то церемонія пройде успішно.

Якщо гарантується, що хоча б один з учасників є чесним, то він обов'язково видалить свою частину токсичних параметрів.

2.3 Побудова доведення в zk-SNARK

В минулому пункті було розглянуто налаштування для zk-SNARK. У якому було обраховано публічні параметри, які потрібні саме для побудови доведення в zk-SNARK. Алгоритм побудови доведення для розрахунку використовує також MLP обчислення. Він є більш комплекснішим, ніж перевірка доведення. Також ключ (prover key) займає більше операційної пам'ятті.

Відповідно до протоколу в [4], сторона P вибирає r , s $\in F_p$ і потім будує значення, що утворюють деведення. Доведення складається з трьох елементів групи:

$$A^{(1)} = AG_1 \text{ , } B^{(2)} = BG_2 \text{ , } i \ C^{(1)} = CG_1 \text{ , де} \tag{2.22}$$

$$A = \alpha + \sum_{i=0}^{m} \alpha_i u_i(x) + r\delta \text{ ,}$$

$$B = \beta + \sum_{i=0}^{m} \alpha_i v_i(x) + s\delta \text{ ,}$$

$$C = \frac{\sum_{i=l+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta} + As + Br - rs\delta \text{ .}$$

Зауважимо, що аналогічно значенням σ_1 та σ_2 в налаштуванні , значення A , B , $C \in F_p$ є також невідомі учасникам, в тому числі і сторонам P і V. Але знаючи значення (2.20) та (2.21) цього достатньо для того, щоб сторона P побудувала значення (2.22). Покажемо, як сторона P будує доведення.

Спочатку будуємо значення $A^{(1)}$:

$$A^{(1)} = AG_1 = \alpha G_1 + \sum_{i=0}^{m} a_i u_i(x) G_1 + r\delta G_1 \ . \tag{2.23}$$

Зауважимо, що αG_1 було знайдено в налаштуванні і є відомим всім учасникам. Значення $r\delta G_1$ сторона P може обчислити:

$$r\delta G_1 = r G_\delta^{(1)} \ , \tag{2.24}$$

де r було зафіксовано, а $G_\delta^{(1)}$ відомо з налаштування.

Обчислимо другий член в рівнянні (2.23):

$$A^{(1)}_{a_1,\ldots,a_m} = \sum_{i=0}^{m} a_i u_i(x) G_1 \ . \tag{2.25}$$

Сторона P спочатку обчислює $u_i(x)G_1$, відповідно до процедури описаної в (2.19), і потім, використовуючи «публічну» інформацію a_i $(i = \overline{0,l})$ та «секретну» інформацію (що знає лише сторона P) a_i $(i = \overline{l+1,m})$, обчислює $A^{(1)}_{a_1,\ldots,a_m}$.

Значення $B^{(2)}$:

$$B^{(2)} = \beta G_2 + \sum_{i=0}^{m} a_i(x) v_i(x) G_2 + s\delta G_2 \ . \tag{2.26}$$

обчислюється так само, використовуючи (2.19), «публічну» інформацію a_i $(i = \overline{0,l})$ та «секретну» інформацію (що знає лише сторона P) a_i $(i = \overline{l+1,m})$ обчислює $a_i v_i(x) G_2$, $i = \overline{0,m}$.

Значення $C^{(1)}$ може бути представлене, як

$$C^{(1)} = \sum_{i=l+1}^{m} a_i G^{(1)}_{\alpha\beta\delta,i} + \frac{h(x)t(x)}{\delta} G_1 + sA^{(1)} + rB^{(1)} - rsG_\delta^{(1)}. \tag{2.27}$$

де $B^{(1)} = BG_1$.

Значення елементу $G^{(1)}_{\alpha\beta\delta,i}$, $i = \overline{l+1,m}$ відоме з налаштування, значення a_i $(i = \overline{l+1,m})$ знає лише сторона P, тож вона легко може

порахувати значення першого члену в рівнянні (2.27). Також сторона P легко може обчислити суму $sA^{(1)} + rB^{(1)} - rsG_{\delta}^{(1)}$.

Для обчислення $\frac{h(x)t(x)}{\delta}G_1$ нагадаємо, що поліном $h(\cdot)$ є відомий лише для сторони P. Запишемо це як:

$$h(y) = h_{n-2}y^{n-2} + ... + h_1 y + h_0 \ , \ h_i \in F_p \ , \ i = \overline{0,n-2} \ .$$

Значення $T_{\delta,i}^{(1)} = x^i \frac{t(x)}{\delta}G_1$ $(i = \overline{0,n-2})$ було отримано при налаштуванні. Звідси сторона P може обчислити значення $\frac{h(x)t(x)}{\delta}G_1$:

$$\frac{h(x)t(x)}{\delta}G_1 = \sum_{i=0}^{n-2} h_i x^i \cdot \frac{t(x)}{\delta}G_1 = \sum_{i=0}^{n-2} h_i T_{\delta,i}^{(1)} \ ,$$

де h_i $(i = \overline{0,n-2})$ відомі лише для сторони P, а значення $T_{\delta,i}^{(1)}$ відомі з налаштування.

Після обрахування CG_1, доведення - побудовано.

Зауважимо, що для обчислення доведення, стороні P не потрібно знати всі значення з векторів $\sigma_1 G_1$ та $\sigma_2 G_2$. Значення, які повинен знати (англ.) $Prover$ називаються (англ.) Proving Key (PK). Дамо опис PK:

$$PK = \{G_{\alpha}^{(1)} \ , G_{\delta}^{(1)} \ , \{X_i^{(1)}\}_{i=0}^{n-1} \ , \{G_{\alpha\beta\delta,i}^{(1)}\}_{i=l+1}^{m} \ , \{T_{\delta,i}^{(1)}\}_{i=0}^{n-2} \ , G_{\beta}^{(2)} \ , G_{\delta}^{(2)} \ , \{X_i^{(2)}\}_{i=0}^{n-1}\}. \tag{2.28}$$

Отже, в даному пункті ми показали, як (англ.) $Prover$ обчислює доведення $(A^{(1)} \ , B^{(1)} \ , C^{(1)})$, після чого викладає його в блокчейн. Для обчислення ми також використовували дані, які були отримані з етапу налаштування, інші дані міг обчислити тільки $Prover$, оскільки тільки він знає значення a_i $(i = \overline{l+1,m})$. Наступним етапом є верифікація побудованого доведення $Verifier$.

2.4 Верифікація в zk-SNARK для QAP

Відповідно до частини «succinct» акроніму це означає, що розмір доведення є доволі малим. Етап верифікації для zk-SNARK є доволі швидким. Сторона V дає відповідь, на основі сформованого доведення з минулого пункту (2.22). Процес верифікації повертає лише один біт: так або ні, чи 0 або 1, і ніякої іншої інформації. Тобто, як раніше вже наголошувалось сторона V не знає нічого про свідчення (англ. witness), яке знає лише сторона P.

Нагадаємо, що сторона V знає загальні параметри, такі як p, $\mathbf{G_1}$, G_1, $\mathbf{G_2}$, G_2 та значення a_i $(i = \overline{0,l})$, і поліноми $u_i(\cdot)$, $v_i(\cdot)$, $w_i(\cdot)$, $i = \overline{0,m}$. Також V знає значення (2.20)-(2.21) обчисленні в налаштуванні і значення $A^{(1)}$, $B^{(1)}$, $C^{(1)}$ виставлені P в блокчейн, в якості доведення.

Відповідно до протоколу [4], сторона V має обчислити значення

$$\begin{gathered} V_{AB} = e(A^{(1)},\, B^{(2)}) \ , \\ V_{\alpha\beta} = e(G_{\alpha}^{(1)},\, G_{\beta}^{(2)}) \text{ (можна предобчислити)}, \\ V_{\alpha\beta\gamma,i} = e(G_{\alpha\beta\gamma,i}^{(1)} \ ,\, G_{\gamma}^{(2)}) \ , \ i = \overline{0,l} \ , \\ V_{C\delta} = e(C^{(1)} \ ,\, G_{\delta}^{(2)}) \ . \end{gathered} \tag{2.29}$$

Потім сторона V перевіряє наступну рівність:

$$V_{AB} = V_{\alpha\beta} \cdot (\sum_{i=0}^{l} a_i V_{\alpha\beta\gamma,i}) \cdot V_{C\delta} \ , \tag{2.30}$$

де операція множення та додавання визначені в скінченому полі F_{r^k}.

Зауважимо, що деякі значення для обчислення ми вже отримали в налаштуванні.

Опишемо, з яких значень складається (англ.) Verification Key (VK):

$$VK = \{G_{\alpha}^{(1)} \ , G_{\beta}^{(2)} \ , G_{\alpha\beta\gamma,i}^{(1)} \ (i = \overline{0,l}) \ , G_{\gamma}^{(2)} \ , G_{\delta}^{(2)}\} \ .$$

З побудовою доведення та перевіркою в zk-SNARK ми завершили. Тепер ми можемо довести коректність для zk-SNARK.

2.5 Коректність в zk-SNARK

Коректність означає, що якщо $Prover$ діє відповідно алгоритму для побудови доведення, що ми описали в секції (2.3), то $Verifier$ завжди прийме його доведення. Інакшими словами, це означає те, що рівняння (2.30) виконується завжди, якщо доведення було створено чесним $Prover$.

Для коректності доведення достатньо показати, що друге рівняння з (2.29) виконується для значень (2.24), (2.26), (2.27).

Відповідно до другого рівняння з (2.29), з використанням білінійної властивості спарювань, ліва частина (2.30) дорівнює

$$V_{AB} = e(A^{(1)}\ ,\ B^{(2)}) = e(AG_1\ ,\ BG_2) =$$
$$e((\alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta)G_1\ ,\ (\beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta)G_2) =$$
$$e(G_1\ ,\ G_2)^{(\alpha+\sum_{i=0}^{m} a_i u_i(x)+r\delta)(\beta+\sum_{i=0}^{m} a_i v_i(x)+s\delta)} =$$
$$e(G_1\ ,\ G_2)^{P_{\text{л.ч.}}}\ \text{, де л.ч. - ліва частина.}$$

Розпишемо ліву частину

$$P_{\text{л.ч.}} = \alpha\beta + \alpha\sum_{i=0}^{m} a_i v_i(x) + \alpha s\delta + \beta\sum_{i=0}^{m} a_i u_i(x) +$$
$$\sum_{i=0}^{m} a_i u_i(x) \cdot \sum_{i=0}^{m} a_i v_i(x) + s\delta\sum_{i=0}^{m} a_i u_i(x) +$$
$$r\delta\beta + r\delta\sum_{i=0}^{m} a_i v_i(x) + rs\delta^2\ . \qquad (2.31)$$

Права частина рівняння (2.30) може бути переписана як

$$V_{\alpha\beta} \cdot (\sum_{i=0}^{l} a_i V_{\alpha\beta\gamma,i}) \cdot V_{C_\delta} = e(G_1,G_2)^{\alpha\beta} \cdot e(G_1,G_2)^{\sum_{i=0}^{l} a_i(\alpha v_i(x)+\beta u_i(x)+w_i(x))} \times$$
$$\times e(G_1,G_2)^{\sum_{i=l+1}^{m} a_i(\beta u_i(x)+\alpha v_i(x)+w_i(x))+h(x)t(x)+As\delta+Br\delta-rs\delta^2} = e(G_1,G_2)^{P_{\text{п.ч.}}}$$

(п.ч. - права частина) ,

де

$$P_{\text{п.ч.}} = \alpha\beta + \sum_{i=0}^{l} a_i(\alpha v_i(x) + \beta u_i(x) + w_i(x)+$$
$$+ \sum_{i=l+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))+$$
$$+As\delta + Br\delta - rs\delta^2 =$$
$$\alpha\beta + \sum_{i=0}^{m} a_i(\alpha v_i(x) + \beta u_i(x) + w_i(x)) + h(x)t(x)+$$

$$+(\alpha s\delta + s\delta \sum_{i=0}^{m} a_i u_i(x) + rs\delta^2) + (\beta r\delta + r\delta \sum_{i=0}^{m} a_i(x) v_i(x) + rs\delta^2) - rs\delta^2 \ .$$

І зараз достатньо показати, що $P_{\text{л.ч.}} = P_{\text{п.ч.}}$.

Після зведення доданків в $P_{\text{л.ч.}}$, $P_{\text{п.ч.}}$, ми розуміємо, що цього достатньо, щоб довести

$$\sum_{i=0}^{m} a_i u_i(x) \cdot \sum_{i=0}^{m} a_i v_i(x) = \sum_{i=0}^{m} a_i w_i(x) + h(x)t(x) \ ,$$

яке виконується відповідно до QAP.

Висновки до розділу 2

У другому розділі було розглянуто побудову zk-SNARK. Спочатку ми описали перехід від R1CS до QAP, показавши цим універсальність

zk-SNARK, оскільки вони можуть бути застосовані до будь-якої обчислювальної задачі. Далі ми описали етап налаштування, як один з найбільш вразливих етапів для витічки інформації. Також було описано алгоритм побудови доведення та верифікації, доведено коректність для zk-SNARK. Було побудовано два ключі: prover key та verification key. Опис SNARK-доведення ми повинні були описати:

– для побудови алгоритму симуляції доведення, нам необхідно зрозуміти: з чого складається доведення, які параметри туди входять, які обчислення необхідні для побудови, все це було описано в даному розділі;

– також SNARK-доведення використовуються у рекурсивних доведеннях, для яких сформульовані властивості у третьому розділі.

3 ВЛАСТИВОСТІ ПАРАМЕТРІВ РЕКУРСИВНОГО SNARK-У ТА СПОСОБИ СИМУЛЯЦІЇ ДОВЕДЕННЯ У SNARK

Використовуючи протокол zk-SNARK, ми ніколи не знаємо правил гри. Правила встановлюються учасниками процедури, що називається довіреним налаштуванням (англ. trusted setup), у якій за допомогою MLP обчислень встановлюються публічні параметри. Але після завершення процедури налаштування, ці правила не можливо перевірити. Ми можемо довіритись даним параметрам. Але якщо ми не брали участь у процедурі налаштування, ми не маємо гарантії, що вони коректні.

У даному розділі ми опишемо алгоритм симуляції доведення, на основі знань певних аргументів. Ми покажемо, які вразливості є у zk-SNARK — це є важливим фактором для подальшого розвитку. Спосіб симуляції залежить від тих значень, що були якимось чином отримані з етапу налаштування. Тому важливо було описати: етап налаштування параметрів, алгоритм побудови доведення, та саму побудову zk-SNARK.

Окрім цього ми опишемо базові поняття рекурсивних SNARK-доведень. Ми доведено твердження про задання трійок, які використовуються у рекурсивних SNARK-ах.

3.1 Симуляція доведення на основі параметрів, отриманих з налаштування SNARK-у

Як зазначалось раніше, всі значення параметрів з σ_1 та σ_2 (див. 2.1) являються невідомими. Це є необхідним, оскільки витік деякої інформації про ці значення призводить до можливості симуляції доведення в zk-SNARK. У випадку з Zcash це означає, що симуляція доведення дає можливість учаснику «генерувати криптовалюту з повітря». Саме тому

етапу налаштування приділяється важлива увага.

Таким чином, якщо хтось знає значення α , β , γ , він може імітувати коректне доведення без жодної іншої інформації.

Більше того, він зможе створити безліч коректних доказів, наступним чином:

Твердження 3.1. *Якщо зловмисник заволодів такими параметрами, як* α , β , δ , a_i *то він може побудувати доведення, використовуючи* ***Алгоритм №3****.*

Алгоритм №3

Симуляція доведення

1) вибираємо довільні значення A , $B \in F_p$;

2) обчислюємо $A^{(1)} = AG_1$ та $B^{(2)} = BG_2$;

3) обчислюємо $C_1^{(1)} = (AB\delta^{-1}) \cdot G_1$;

4) обчислюємо $C_2^{(1)} = (\alpha\beta\delta^{-1}) \cdot G_1$;

5) використовуючи значення $X_j^{(1)}$, $j = \overline{0,n-1}$ та техніку, що була була описана в пункті налаштування, обчислюємо:

$$u_i(x)G_1 \text{ , } v_i(x)G_1 \text{ та } w_i(x)G_1 \text{ , } i = \overline{0,l} \text{ ;}$$

6) обчислюємо:

$$C_3^{(1)} = \delta^{-1} \sum_{i=0}^{l} a_i(\beta u_i(x)G_1 + \alpha v_i(x)G_1 + w_i(x)G_1) = \frac{\sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta} G_1 \text{ ;}$$

7) обчислюємо $C^{(1)} = C_1^{(1)} + C_2^{(1)} + C_3^{(1)}$;

8) викладаємо трійку $A^{(1)}$, $B^{(2)}$, $C^{(1)}$ в блокчейн, як доведення .

Ми справедливо можемо стверджувати, що дана трійка буде проходити перевірку, і буде виконуватись рівняння (2.30) .

3.2 Симуляція доведення на основі параметрів, отриманих під час побудов доведення

Для даного способу введемо деякі позначення:

– C - зловмисник/хакер, особа, яка за допомогою обману бажає заволодіти даними, які їй не належать;

– D - данні, якими заволодів C в результаті певних обчислень або певних атак;

Розглянемо наступну ситуацію, було виконано етап налаштування та побудовано два доведення (позначимо proof 1 та proof 2). В результаті чого C дізнався деяку інформацію, з кожного доведення:

1) При побудові proof 1, C дізнався значення A та r,$s \in F_p$ позначимо, як D_1 ;

2) При побудові proof 2, C дізнвся значення B та $r^{'}$,$s^{'} \in F_p$ позначимо, як D_2 ;

Використовуючи дані D_1 та D_2 , C може обчислити секретні параметри, що були використані в налаштуванні, а саме α та β . Покажемо, як він це може зробити.

Алгоритм №4

1) C обраховує δ наступним чином: оскільки G_1 - це елемент групи, для нього існує обернений, такий що $G_1 G_1^{-1} = G_1^{-1} G_1 = e$, де e - одиничний елемент групи. Значення δG_1 ми знаємо з налаштування, тому що його було викладено в блокчейн. Використовуючи означення одиничного елементу, домножуємо δG_1 на G_1^{-1} отримуємо значення δ.

2) Знаючи D_1 та D_2 , розпишемо значення A , B відповідно до алгоритму побудови доведення:

$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + \delta r \ ;$$

$$B = \beta + \sum_{i=0}^{m} a_i v_i(x) + \delta s' \ ;$$

3) Значення сум $f = \sum_{i=0}^{m} a_i u_i(x)$ та $k = \sum_{i=0}^{m} a_i v_i(x)$ C передобчислює, відповідно до (2.19).

4) На основі отриманих вище даних, C обраховує параметри α , β :

$$\alpha = A - f - \delta r \ ;$$
$$\beta = B - k - \delta s' \ ;$$

5) Знаючи α , β зловмисник обраховує C.

6) Обчисливши

$$A^{(1)} = AG_1 \ , \ B^{(2)} = BG_2 \ , \ C^{(1)} = CG_1 \ ,$$

зловмисник викладає доведення в блокчен, яке є коректним.

Отже, у даному способі ми описали, як зловмисник може зімітувати доведення, не бравши участі у налаштуванні або пропустивши цей етап. Дані D_1 та D_2 зловмисник отримує нечесним способом під час побудови доведення чесним Prover-ом.

Зауваження 1. Імітувати доведення можливо також без участі в налаштуванні, на основі тих даних, що можна нечесним шляхом отримати з побудови доведення.

Зауваження 2. Доведення, що побудоване в **Алгоритм №4** валідне лише для того налаштування, яке використовувалось для побудови proof 1 та proof 2.

3.3 Недоліки алгоритму побудови доведення

Рахується, що найбільш вразливим місцем в побудові zk-SNARK є налаштування. Але окрім цього вразливим місцем також є елементи A , B , C , які формує Prover для доведення. Оскільки дізнаючись їх, є

можливість відтворити секретні параметри з налаштування та будувати proof з повітря. Розглянемо декілька можливих варіантів обходу цієї проблеми, які можуть прийти на думку:

1) Проводити етап налаштування після кожного побудованого доведення. Звичайно, це найгірший варіант для блокчейну, оскільки, як ми говорили раніше, час та пам'ять є критичними параметрами.

2) Проводити етап налаштування періодично. Наприклад, зафіксувати значення $T = 100$, це означає, що після 100-го побудованого доведення, блокчейн перебудовується під новий етап налаштування. Які недоліки в данній пропозиції? А недоліки в тому, що блокчейн система не дає гарантії, що в проміжку між першим і 100-им доведенням, не буде ніякої витічки інформації.

Постає питання, як можна виправити проблему витічки інформації під час побудови доведення. Відповідь на дане питання є пункт (2.6). Так, це рекурсивні zk-SNARK. Для рекурсивних zk-SNARK нам не потрібно виставляти в блокчейн елементи доведення $A^{(1)}$, $B^{(2)}$, $C^{(1)}$ ми просто доводимо знання цих елементів.

3.4 Рекурсивні SNARKs

В рекурсивних SNARKs Prover не відкриває елементи $A^{(1)}$, $B^{(2)}$, $C^{(1)}$ з (2.22), але замість цього він доводить знання цих елементів. Для цього він переписує рівняння верифікатора (2.30) у форму QAP над полем F_r. Він може також додати деякі рівняння в QAP, які являють собою транзакції, або смарт контракти, або щось схоже. Оскільки всі рівняння нового QAP виконуються в полі характеристики r , йому потрібні деяки групи порядку r з спарюванням для побудови відповідного SNARK. Це означає, що йому потрібна інша трійка $\mathbf{TR}^{'} = (\mathbf{G_1^{'}}$, $\mathbf{G_2^{'}}$, $\mathbf{G_{TG}^{'}})$ з $|\mathbf{G_1^{'}}| = |\mathbf{G_2^{'}}| = |\mathbf{G_{TG}^{'}}| = r$, де $\mathbf{G_1^{'}}$ це деяка підгрупа еліптичної кривої $E(F_q)$ над простим полем F_q; $\mathbf{G_2^{'}}$ це деяка підгрупа еліптичної кривої $E(F_{q^s})$ над розширенням F_{q^s}; і $\mathbf{G_{TG}^{'}}$ це підгрупа $F_{q^s}^*$; і відповідне відображення.

Далі, коли ми побудували відповідний SNARK, використовуючи $\mathbf{TR}' = (\mathbf{G}_1' , \mathbf{G}_2' , \mathbf{G}_{\mathbf{TG}}')$, нам потрібна інша трійка для перевірки відповідного доведення. Чудовою ідеєю для цього є використання першої нашої трійки $\mathbf{TR} = (\mathbf{G_1} , \mathbf{G_2} , \mathbf{G_{TG}})$. Але це можливо лише при умові, якщо $q = p$.

Тож, в трійці $\mathbf{TR}' = (\mathbf{G}_1' , \mathbf{G}_2' , \mathbf{G}_{\mathbf{TG}}')$ повинні бути наступні групи: $\mathbf{G}_1'$ це деяка підгрупа еліптичної кривої $E(F_p)$ над простим полем F_p; $\mathbf{G}_2'$ це деяка підгрупа еліптичної кривої $E(F_{p^s})$ над розширенням F_{p^s}; і $\mathbf{G}_{\mathbf{TG}}'$ це підгрупа $F_{p^s}^*$; і $|\mathbf{G}_1'| = |\mathbf{G}_2'| = |\mathbf{G}_{\mathbf{TG}}'| = r$.

Для рекурсивного SNARK нам потрібно дві трійки, $\mathbf{TR} = (\mathbf{G_1} , \mathbf{G_2} , \mathbf{G_{TG}})$ та $\mathbf{TR}' = (\mathbf{G}_1' , \mathbf{G}_2' , \mathbf{G}_{\mathbf{TG}}')$, таких що:

– в $\mathbf{TR} = (\mathbf{G_1} , \mathbf{G_2} , \mathbf{G_{TG}})$: $\mathbf{G_1}$ це деяка підгрупа еліптичної кривої $E(F_r)$ над простим полем F_r; $\mathbf{G_2}$ це деяка підгрупа еліптичної кривої $E(F_{r^k})$ над розширенням F_{r^k}; і $\mathbf{G_{TG}}$ це підгрупа $F_{r^k}^*$; і виконується рівність $|\mathbf{G_1}| = |\mathbf{G_2}| = |\mathbf{G_{TG}}| = p$;

– в $\mathbf{TR}' = (\mathbf{G}_1' , \mathbf{G}_2' , \mathbf{G}_{\mathbf{TG}}')$: $\mathbf{G}_1'$ це деяка підгрупа еліптичної кривої $E(F_p)$ над простим полем F_p; $\mathbf{G}_2'$ це деяка підгрупа еліптичної кривої $E(F_{p^s})$ над розширенням F_{p^s}; і $\mathbf{G}_{\mathbf{TG}}'$ це підгрупа $F_{p^s}^*$; і виконується рівність $|\mathbf{G}_1'| = |\mathbf{G}_2'| = |\mathbf{G}_{\mathbf{TG}}'| = r$;

Зауваження: $\mathbf{TR} = (\mathbf{G_1} , \mathbf{G_2} , \mathbf{G_{TG}})$ та $\mathbf{TR}' = (\mathbf{G}_1' , \mathbf{G}_2' , \mathbf{G}_{\mathbf{TG}}')$ повинні бути різними трійками, тому що:

$$\mathbf{G_{TG}} < F_{r^k}^* \Rightarrow |\mathbf{G_{TG}}| \, |r^k - 1 \Rightarrow |\mathbf{G_{TG}}| \neq r \ .$$

Сьогодні рекурсивні SNARKs використовуються в протоколі (англ.) Coda protocol[7]. Хочеться також зауважити те, що рекурсивні SNARK-доведення можуть бути побудованні, використуваючи властивість нульового розголошення, так і не ввикористовуючи властивість. В протоколі Coda не використовується аспект нульового розголошення.

3.5 Теорема про triplets в рекурсивних SNARK-ів

Теорема 3.1. *Нехай* $\mathbf{TR} = (G_1 , G_2 , G_{TG})$ *і* $\mathbf{TR}' = (G_1' , G_2' , G_{TG}')$ *- два тріплети для рекурсивного SNARK, з базовими полями полями* F_q *та* F_p *, відповідно, і при цьому* $|G_1| = |G_2| = |G_{TG}| = p$ *,* $|G_1'| = |G_2'| = |G_{TG}'| = r$*. Тоді, якщо SNARK є стійким, то наступні твердження справедливі:*

1) $\mathbf{TR}$ *та* $\mathbf{TR}'$ *- обов'язково різні тріплети;*

2) порядки груп точок еліптичних кривих у тріплетах - обов'язково прості (тобто кофактори дорівнюють одиниці).

Доведення.

1) Припустимо, що ми можемо побудувати один тріплет $\mathbf{TR}$, який можна використовувати для рекурсивного SNARK. Нехай базовим полем для цього SNARK-у є поле F_r. Тоді, для того, щоб побудувати рекурсію, повинні виконуватись наступні умови:

$$\text{Існує крива } \varepsilon_1(F_r), \text{ така що } |\varepsilon_1(F_r)| = l \cdot r \ , \forall \ l \in N. \tag{3.1}$$

Дійсно, за цієї умови

$$G_1 < \varepsilon_1(F_r), \ |G_1| = r, \tag{3.2}$$

і ми можемо використовувати групу G_1 для перевірки SNARK-доведення тверджень про елементи поля F_r. Але за теоремою Хассе,

$$|r + 1 - l \cdot r| \leq 2\sqrt{r}, \text{ або } (\sqrt{r} - 1)^2 \leq l \cdot r \leq (\sqrt{r} + 1)^2. \tag{3.3}$$

Покажемо, що при $l \geq 2$, рівність (3.3) не виконується. Дійсно, з правої нерівності у (3.3) випливає

$$l \leq \frac{(\sqrt{r} + 1)^2}{r} \leq \frac{r + 2\sqrt{r} + 1}{r} = 1 + \frac{2}{\sqrt{r}} + \frac{1}{r} < 2, \text{ при } r > 4.$$

Далі, з лівої нерівності у (3.3) випливає

$$l \geq \frac{(\sqrt{r}-1)^2}{r} \geq \frac{r-2\sqrt{r}+1}{r} = 1 - \frac{2}{\sqrt{r}} + \frac{1}{r} \geq 0.$$

Отже, оскільки $l \in N$, то лишається єдина можливість $l = 1$ і $|\varepsilon_1(F_r)| = r$. Але, крім того

$$|\varepsilon_1(F_r)| = |G_{TG}| = r \text{ і } G_{TG} < F_r^*.$$

А тоді, за теоремою Лагранжа повинна виконуватись умова ($k \in N$)

$$r \mid r^k - 1,$$

що не може бути. Отже, тріплети $\mathbf{TR}$ та $\mathbf{TR}'$ повинні бути різними.

2) Нехай для перевірки SNARK - доведення, побудованого з використанням елементів поля F_r, використовується тріплет $\mathbf{TR}'$, і навпаки: для перевірки SNARK - доведення, побудованого з використанням елементів поля F_p , використовується тріплет $\mathbf{TR}$. Тоді повинні виконуватись наступні умови:

$$G_1 < \varepsilon(F_r),\ |G_1| = p\ ;$$
$$G_1' < \varepsilon(F_p),\ |G_1'| = r\ ;$$
$$\text{тобто } |\varepsilon(F_r)| = l \cdot p, \text{ для деякого } l \in N,$$
$$\text{і } |\varepsilon(F_p)| = s \cdot r, \text{ для деякого } s \in N.$$

Але, за теоремою Хассе,

$$(\sqrt{r}-q)^2 \leq l \cdot p \leq (\sqrt{r}+1)^2,$$

звідки

$$l \cdot p \leq (\frac{\sqrt{p}+1}{\sqrt{s}}+1)^2, \text{ або}$$
$$l \leq (\frac{1+\frac{1}{\sqrt{p}}}{\sqrt{s}}+\frac{1}{\sqrt{p}})^2 \leq 1.$$

Аналогічно, з нерівності $(\sqrt{p}-1)^2 \leq s \cdot r \leq (\sqrt{p}+1)^2$ отримаємо $r \leq 1$. Теорему доведено. □

Висновки до розділу 3

У даному розділі були отримані наступні результати:

1) побудовано алгоритм симуляції доведення на основі даних з налаштування;

2) побудовано алгоритм симуляції доведення на основі даних, що отримані під час побудови інших доведень;

3) сформульовано властивості для рекурсивних SNARK-доведень;

4) доведено властивості рекурсивних SNARK-доведень.

ВИСНОВКИ

У даному дослідженні ми привели опис протоколу доведення без розголошення та його форми, інтерактивну та неінтерактивну. Ми розглянули налаштування, побудову, верифікацію SNARK-доведення. Був приведений огляд рекурсивних доведень.

Як результат дослідження я показав, що симуляцію доведення можна провести двома способами. Перший через параметри, що були отримані з етапу налаштування. Другий через параметри, що були отримані в результаті маніпуляцій під час побудови доведення іншими користувачами мережі. Також, було наголошено на тому, що пристальну увагу потрібно приділяти не тільки етапу налаштування, а й алгоритму побудови доведення. Під час роботи було сформульовано та доведено властивості для рекурсивних SNARK-доведень.

Побудовані алгоритми симуляції доведення можна використовувати для подальшого розвитку протоколу zk-SNARK[4].

Сформовані властивості для рекурсивних SNARK-доведень можна використовувати для подальшого розвитку протоколу Coda[7].

ПЕРЕЛІК ПОСИЛАНЬ

1. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexityof interactive proof systems.SIAM Journal on computing, 18(1):186–208, 1989.

2. Sanjam Garg: Non-Interactive Zero-Knowledge (NIZK) and the Hidden-Bit Model
https://people.eecs.berkeley.edu/~sanjamg/classes/cs276-fall14/scribe/lec10.pdf

3. Zcash: SNARK
https://z.cash/technology/zksnarks/

4. Jens Groth On the Size of Pairing-based Non-interactive Arguments
https://eprint.iacr.org/2016/260.pdf

5. Alisa Pankova. Succinct non-interactive arguments from quadraticarithmetic programs. Technical report, Technical report, University of Tartu,Cybernetica AS, 2013.

6. Rudolf Lidl and Harald Niederreiter.Finite fields, volume 20.Cambridge university press, 1997.

7. Coda protocol
https://codaprotocol.com/docs